```
/*
  Braxton Friend
  Encoding and Decrypting Project 3
  CS101
  11/7/2018 - 11/12/2018
*/

import java.util.*;
import java.io.*;
public class EncodingAndDecoding
{
 /* s
   Algorithim for Main(Args)
   inFile <-- File object for command args[0]
   keyFile  <--  File object for command args[1]
   outFile  <-- File object for command args[2]
   decodeEncode <-- String Object for command args[3]
   out <-- PrintStream Object for printing to Output File
   fileScan <-- Scanner object for reading input File
   keyScan <-- Scanner object for reading Key File
   key <-- String variable that stores the next line of Key file and converts to lowercase
   switch : gets the first value of decodeEncode and sets it to lower case to decide if encoding or decoding
   case e : call encode
   case d : call decode
   default
 */

 /*
 Variable/Constant   Type     Purpose
   args           String[]    parameter, unused
   inFile         File        gets input File location from command line
   keyFile        File        gets key file location from command line
   outFile        File        gets output file location from command line
   out            PrintStream prints the output to output file
   fileScan       Scanner     reads the input file
   keyScan        Scanner     reads the key
   key            String      makes the first line of key a string and converts it to lowercase

 */
   public static void main(String [] args) throws Exception
   {
      File inFile = new File(args[0]);
      File keyFile = new File(args[1]);
      File outFile = new File(args[2]);
      String decodeEncode = new String(args[3]);
      PrintStream out = new PrintStream(outFile);
      Scanner fileScan = new Scanner(inFile);
      Scanner keyScan = new Scanner(keyFile);
      String key = keyScan.nextLine().toLowerCase();

      switch (decodeEncode.toLowerCase().charAt(0))
      {
         case 'e' : encode(fileScan, key, out);
            break;
         case 'd' : decode(fileScan, key, out);
            break;
         default :
      }



   }//main

 /*
   Algorithim encode(Scanner input, String key, PrintStream out)
   keyPosition <-- 0
   while( input.hasNextLine())
   String inputLine <-- get next Line of input
```

```
   for(int inPosition <-- 0; inPosition < inputLine.length(); inPosition++)
      if(keyPosition >= key.length())
         keyPosition <-- 0
       keyChar <-- get the char at keyPosition
       char inChar <-- get the char at inPosition
       if(Character.isLetter(inputLine.charAt(inPosition)) && Character.isLetter(charKey))
         inChar <-- get the char at inPosition
       upperCase <-- make true if character from inChar is upper case
         if(the character is an upper case)
            set the character to lower case
      numberFromKey <-- keyChar - 'a'
      numberFromInput <-- inChar - 'a'
      encodeNumber <-- numberFromInput - numberFromKey
      if encodeNumber < 0
         encodeNumber += 26
      outputEncode =  (char) ('a' + encodeNumber)
      if (the character was upper case)
        set the character back to upper case
      print the character outputEncode formed
      increment the keyPosition++
      else
         print inChar
         increment the keyPosition++

    print new line in outputFile

 */


  /*
Variable/ Constant   Type              Purpose
input                scanner           parameter, used as decryptAndEncrypt in main method
key                  string            parameter, used as keyString in main method
out                  printStream       paramter, used as out in main method
keyPosition          int               position of each character in key
inputLine            string            converts the next line of input into string
outputEncode         char              the value to print at the end to the output file
inPosition           int               position of each character in nextInput
keyChar              char              character of key at keyPosition
inChar               char              character of inputLine at inPosition
numberFromKey        int               turns the character from keyChar into a number value
numberFromInput      int               turns the character from inChar into a number value
encodeNumber         int               the number value of the decoded character
upperCase            boolean           to check if inChar is upper case

 */
 //decode is the same as encode, just with subtraction to finish the decoding
   public static void decode(Scanner input, String key, PrintStream output)
   {
      while(input.hasNextLine())
      {
         int keyPosition = 0;
         String inputLine = input.nextLine();
         for(int inPosition = 0; inPosition < inputLine.length(); inPosition++)
         {
            if (keyPosition >= key.length())
            {
               keyPosition = 0;
            }
            char keyChar = key.charAt(keyPosition);
            char inChar = inputLine.charAt(inPosition);
            if(Character.isLetter(inputLine.charAt(inPosition)) && Character.isLetter(keyChar))
            {
               inChar = inputLine.charAt(inPosition);
               boolean upperCase = Character.isUpperCase(inChar);
               if (upperCase)
               {
                  inChar = Character.toLowerCase(inChar);
```

```
137                }
138                int numberFromKey = keyChar - 'a';
139                int numberFromInput = inChar - 'a';
140                int encodeNumber = numberFromInput - numberFromKey;
141                if(encodeNumber < 0)
142                encodeNumber += 26;
143                char outputEncode = (char)('a' + encodeNumber);
144                if(upperCase)
145                   outputEncode = Character.toUpperCase(outputEncode);
146                output.print(outputEncode);
147                keyPosition++;
148             }
149             else
150             {
151                output.print(inChar);
152                keyPosition++;
153             }
154          }
155          output.print("\n");
156       }
157    }
158    /*
159 Algorithim encode(Scanner input, String key, PrintStream out)
160 keyPosition <-- 0
161 while( input.hasNextLine())
162 String inputLine <-- get next Line of input
163    for(int inPosition <-- 0; inPosition < inputLine.length(); inPosition++)
164       if(keyPosition >= key.length())
165          keyPosition <-- 0
166        keyChar <-- get the char at keyPosition
167        char inChar <-- get the char at inPosition
168        if(Character.isLetter(inputLine.charAt(inPosition)) && Character.isLetter(charKey))
169          inChar <-- get the char at inPosition
170        upperCase <-- make true if character from inChar is upper case
171          if(the character is an upper case)
172             set the character to lower case
173          numberFromKey <-- keyChar - 'a'
174          numberFromInput <-- inChar - 'a'
175          encodeNumber <-- numberFromInput + numberFromKey
176          if encodeNumber > 25
177             encodeNumber -= 26
178          outputEncode =  (char) ('a' + encodeNumber)
179          if (the character was upper case)
180             set the character back to upper case
181          print the character outputEncode formed
182          increment the keyPosition++
183          else
184          print inChar
185          increment the keyPosition++
186
187     print new line in outputFile
188
189    */
190
191
192    /*
193    Variable/ Constant      Type            Purpose
194    input                scanner        parameter,  used as decryptAndEncrypt in main method
195    key                  string         parameter,  used as keyString in main method
196    out                  printStream    paramter,   used as out in main method
197    keyPosition          int            position of each character in key
198    inputLine            string         converts the next line of input into string
199    outputEncode         char           the value to print at the end to the output file
200    inPosition           int            position of each character in nextInput
201    keyChar              char           character of key at keyPosition
202    inChar               char           character of inputLine at inPosition
203    numberFromKey        int            turns the character from keyChar into a number value
204    numberFromInput      int            turns the character from inChar into a number value
```

```
   encodeNumber         int            the number value of the encoded character
   upperCase            boolean        to check if inChar is upper case

   */



   public static void encode(Scanner input, String key, PrintStream output)
   {
      while(input.hasNextLine())
      {
         int keyPosition = 0;
         String inputLine = input.nextLine();
         for(int inPosition = 0; inPosition < inputLine.length(); inPosition++)
         {
            if (keyPosition >= key.length())
            {
               keyPosition = 0;
            }
            char keyChar = key.charAt(keyPosition);
            char inChar = inputLine.charAt(inPosition);
            if(Character.isLetter(inputLine.charAt(inPosition)) && Character.isLetter(keyChar))
            {
               inChar = inputLine.charAt(inPosition);
               boolean upperCase = Character.isUpperCase(inChar);
               if (upperCase)
               {
                  inChar = Character.toLowerCase(inChar);
               }
               int numberFromKey = keyChar - 'a';
               int numberFromInput = inChar - 'a';
               int encodeNumber = numberFromKey + numberFromInput;
               if(encodeNumber > 25)
                  encodeNumber -= 26;
               char outputEncode = (char)('a' + encodeNumber);
               if(upperCase)
                  outputEncode = Character.toUpperCase(outputEncode);
               output.print(outputEncode);
               keyPosition++;
            }
            else
            {
               output.print(inChar);
               keyPosition++;
            }
         }
         output.print("\n");
      }
   }

   //classMethod

   /*
     Note that you will most likely have multiple class methods, so repeat
     this algorithm, data table, and code section as often as needed.
     The return type does not have to be void (int and boolean are likely).
     Also note that the parameter name and types will be different from
       String [] args.
   */

}//class
```